Sensor Fusion For Autonomous Outdoor Navigation Using Neural Networks

Ian Lane Davis

CMU-RI-TR-95-05

Carnegie Mellon University
The Robotics Institute

THE ROBOTICS INSTITUTE

Technical Report

Sensor Fusion For Autonomous Outdoor Navigation Using Neural Networks

Ian Lane Davis

CMU-RI-TR-95-05

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

January, 1995

This work has been supported in part by a National Science Foundation Graduate Research Fellowship. The equipment used is provided in part by DACA76-89-C0014, Topographic Engineering Center, Perception for Outdoor Navigation and DAAE07-90-C-R059, TACOM, CMU Autonomous Ground Vehicle Extension.

Sensor Fusion For Autonomous Outdoor Navigation Using Neural Networks

Ian Lane Davis

CMU-RI-TR-95-05

Abstract

For many navigation tasks, a single sensing modality is sufficiently rich to accomplish the desired motion control goals; for *practical* autonomous outdoor navigation, a single sensing modality is a crippling limitation on what tasks can be undertaken. In the research detailed in this paper, we open the door for a whole new suite of real-time autonomous navigation tasks previously unattainable. Using neural networks, including a neural network paradigm particularly well suited to sensor fusion, and Carnegie Mellon University's HMMWV (High Mobility Multi-Wheeled Vehicle) off-road military ambulance, we have successfully performed simulated and real-world navigation tasks that required the use of multiple sensing modalities.

Table of Contents

1.0 Introduction

The autonomous land vehicle group at CMU has for years been studying autonomous navigation for wheeled vehicles using the Navlab I and the HMMWV (Navlab II). Some of the most successful work to come out of the group has included road following using CCD (charge coupled device) cameras [Pomerleau92, Jochem93] and cross country navigation using a scanning laser range finger [Langer93, Kelly93]]. The long term goal of each of these projects has been to engineer a vehicle capable of performing fully autonomous tasks in a given environment (e.g. on a highway or in the wilderness).

1.1 Limitations of the state of the art

Earlier systems have had some obvious limitations. Primarily, a single CCD camera is sufficient for road following, but insufficient for extended road driving tasks, and a laser range finder is sufficient for obstacle avoidance, but insufficient for cross country (XC) navigation in "interesting" terrains. Road following capability alone will not allow a robot to drive on a roadway on which there are obstacles, and simple range-based obstacle avoidance can be both inefficient and ineffective in the presence of vegetation, water, or other "unusual" obstacles.

One direction of research to explore is the augmentation of current capabilities with "top-down" information. For road following this might mean using limited access roads and having all vehicles on the road signal each other about their whereabouts. The obvious difficulty is dealing with unexpected obstacles, such as fallen branches, animals, and even stalled or damaged cars. For cross country navigation, top-down augmentation might involve the use of maps in coordination with some sort of global positioning/intertial navigation system. Such a system is limited in previously unexplored regions. Top-down augmentation in either case is subject to failures due to communications breakdowns or incorrect information which cannot be directly confirmed via available local sensing.

1.2 Fully self-contained autonomous navigation

In both the road following and the cross country navigation tasks, as well as in many other tasks for which the use of robots is needed, such as space exploration, *fully self-contained* navigation is either very desirable or entirely necessary. The techniques which have worked so well to this date for the limited scenarios (in ALVINN and Ranger) have been "bottom-up" techniques, which we can define as techniques which take the rawest of locally available data and turn this into the necessary motion control commands for the task. Inherent in this concept is rich sensing, and for the tasks at hand the broadest way to expand the sense-space of the robot is via additional local sensing modalities.

Thus, our goal is to integrate multiple sensors to achieve capabilities not previously attainable in autonomous navigation systems. These capabilities are achievable with "bottom-up" techniques such as those described later. For the reasons mentioned above, using such an approach is desirable and often necessary.

1.3 Prior work & our approach: the MAMMOTH system

The approach taken in our research is informed by much of the work in cross country naviation [Daily88, Brumitt92, Langer93, and Kelly93], work in neural networks for navigation [Pomerleau92, Meeden93, Marra88, and Wright89], and work in modular neural networks [Jacobs91, Smieja92, Hampshire89, Fahlman91, Gluck93, and Jochem93].

Our current experiments with sensor fusion use backpropagation neural networks [Rumelhart86]. The basic elements of a neural network are nodes and connections. Nodes have activation levels, and connections have weights. In a typical network, there are a number of input nodes, each of which has some level of activation (these can represent the pixels in an image with their activation level being their intensity in the image). Most often, there is a next level of nodes, called hidden units. Each hidden unit is connected to each input unit (unit and node are used interchangeably) by a connection with a given weight. The activation level of a hidden node is the sum of the activation of each node it's connected to multiplied by the weight of the connection. The activation is limited to being between -1.0 and 1.0 (or zero and one) by a sigmoidal function. Hidden units come in layers usually and often there is only a single layer which connects to a group of nodes known as the output nodes. The network learns by having some pattern put into the input nodes and adjusting the weights of all of the connections until the output nodes show the desired output pattern.

Why do we choose to use neural networks? For an excellent discussion of the suitability of neural networks for the general navigation task see [Pomerleau92]. The primary issue is that we do not necessarily know all of the features in sensor space which affect our navigation decisions. The fallout is that if we use a model-based system of navigation we would need to explicitly take *all* of the features into account and experimentally (or arbitrarily) set the relative importance of each feature. We cannot expect performance in the presence of a feature rich environment such as the real outdoor world, and we certainly can expect to go back to the drawing boards in any new environment. Neural networks when properly trained can automatically select and weight the most important features of an environment, with the added bonus of being able to tackle a radically different environment with just a change of training data.

The architectures of the networks we are examining in this work are of two flavors: monolithic networks and modular networks. In the typical monolithic neural network approach, we give the network all of the sensor data as inputs and allow the network to develop internal representations which allow it to perform adequately within the context of the experiences we expose it to. We might even train the network in different situations to allow us a broader range of "safe" navigation terrains. A monolithic network may be seen in Figure 1, on page 3.

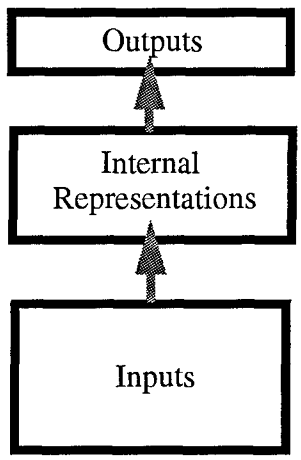

FIGURE 1. A typical monolithic neural network in which the inputs feed forward into internal representations (developed during training) which in turn feed forward into the outputs. For driving, inputs might be the pixels of a video image and the outputs might be some representation of a steering direction.

One potential shortcoming of this approach is that, while in a model-based system we can't guarantee performance over all possible inputs, here, we cannot *guarantee* adequate performance in any situation, although we might *expect* adequate performance in most cases. This is because we do not always know how to interpret the internal representations that the neural network generates in complicated scenarios.

What if we know that some feature in the sensor space is useful for navigation? For example, we know that the edges of the road are tremendously good clues to use in on-road navigation. A typical neural paradigm might result in a network learning to find the edges of the road. Or it might not. But even if it does, we might have directed the learning towards less obvious but also important features if we first shared our a priori knowledge of the task and sensor domain. Exactly how much of our resources (time, memory, computing power, patience) are we devoting to learning what we already know?

This work uses monolithic neural networks for navigation, but the above concern, as well as the concern that monolithic neural networks might not scale easily as we increase the demands on our system in terms of input space and output (actuation) space, has led us to examine modular neural networks as an alternative. The modular architecture we use attempts to address the issue of integrating a priori knowledge and hints that it may prove to be more flexible as the complexity of the tasks increase. Our modular system is the Modular Architecture Multi-Modal Theory network (MAMMOTH).

A MAMMOTH system consists of two segments, the feature level and the task level. The feature level is a set of Feature Neural Networks (FeNN) trained to recognize hand-picked features in whatever sensor modality serves as their input source. Thus, there might be one or more networks in the feature level devoted to the color video modality and each of these would be trained to recognize features in color video space, and their hidden units would be encoding these features in color video space. Other networks in this level might address features in laser range finder sensor space or in the input space of any modality we care to employ.

On top of this feature level is the task dependent network which unites the feature networks, the Task Dependent Neural Network (Task Net). The task level uses the encoded information from the feature level networks as inputs to a larger network trained to perform the navigation task. These encoded feature inputs can be supplemented with task-related inputs. For a sketch of a MAMMOTH network see Figure 2, on page 5.

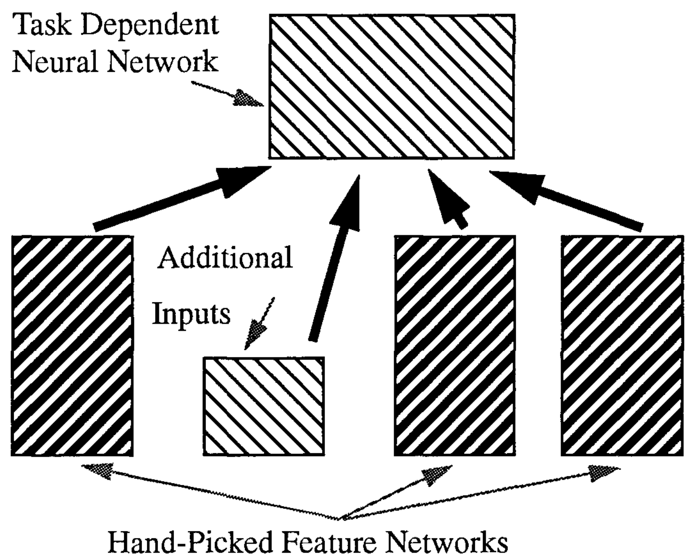

FIGURE 2. General MAMMOTH architecture

This paper represents the first use of both the monolithic neural network and the MAMMOTh network on the outdoor navigation tasks described later. For a more detailed description of the MAMMOTH paradigm and of its advantages, see [Davis93b].

2.0 The task at hand

2.1 The robot

Our specific task is to achieve autonomous navigation with the Navlab II, or HMMWV, four-wheel-drive military ambulance (see Figure 4, on page 7). In our early experiments we are focussing on a "wander" behavior. We want the vehicle to wander around the terrain and not get into danger. This involves avoiding obstacles, but not heading to particular goals (this will be added later). We want our system to choose an appropriate steering direction. At this point in time, the vehicle's throttle is not actuated, so the safety driver (a person in the vehicle who can take control at any moment) controls the gas pedal and brakes.

The HMMWV (pronounced "humvee") has all of its computing on-board, a gas generator for power, a battery back-up for the computing power, an air-conditioned computing cage for custom cards (often for sensing) as well as the workstations, and a whole suite of sensors, including a color video camera, a scanning laser range sensor, and a multibaseline stereo rig. The vehicle weighs 6 tons and can be driven at up to 70 miles per hour. For work on the vehicle, three SPARC™ 10 computers, named Moe, Larry, & Curly, handle all of the processing.

FIGURE 3. The HMMWV

Most of our work is tested in one of two environments: at 2 kilometer by 2 kilometer outdoor testing site on top of an old slag heap or on a square kilometer of a "virtual" 2.5D world.

The slag heap in the real world has sections that range from very sparsely vegetation with dry grass to intraversible due to trees and thick bushes. There are natural and man-made ridges of earth, man-made mounds of earth, cliffs, large semi-permanent puddles, and remnants of the foundations of a few old buildings. A dirt access road runs through a large section of the slag heap (looping around to almost 4k total length). A rough estimate would be that less than half of the total surface area of the slag heap is traversible with the HMMWV (i.e. the vehicle itself is capable of safely moving over it). Most current systems are very limited in which sections they can maneuver in because they do not use multiple sensors (and correspondingly cannot deal well with vegetation).

The simulator, called Alsim [Kelly93], used by the cross country navigation group at CMU consists of a map with elevation and color (RGB) in each cell. The initial elevations are generated with a triangle based fractal, and coloration is based on a similar fractal. This creates a world with varying elevation and patches of "vegetation" (a.k.a. green stuff). We can add onto this world a variety of geometric shapes, such as gaussians, cylinders, boxes, pyramids, ridges, and cones, in any desired color. Furthermore, dirt or paved roads (with a yellow line running down the middle) can be added by selecting a number of points in 3D through which the road will run and to which a spline is fit and the road filled in on the map.

In both the real world and in Alsim, the sensors used are a steering encoder, the color video camera, and the ERIM laser range finder. Sample images from the real and simulated imaging sensors are shown below (Figure 4, on page 7).

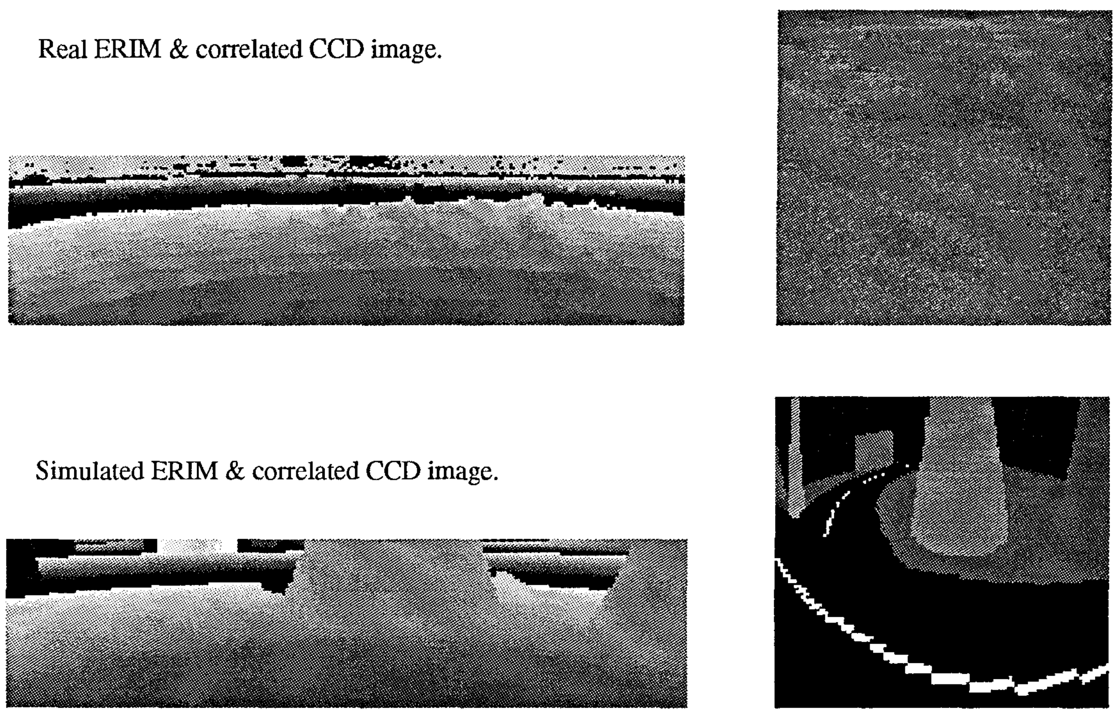

FIGURE 4. Images from the real world and from Alsim In ERIM images, intensity encodes depth.

3.0 System issues

The most recent experiments fall into the obvious categories: simulation in Alsim, and real world experimentation. In both cases, but especially in the real world, speed of processing is important, and this has affected the design of the control system in several ways.

3.1 Image resolution

First, while in simulation we can generate both range and color video images on the same computer and call them synchronized, on the real vehicle this is impossible. For our purposes the images do not have to be perfectly synchronized, but since the ERIM operates at 2Hz, digitizing sequentially would mean more than a half second of travel between starting the laser range scan and starting the color camera digitization (and on a more practical level, both digitizers will not operate properly on the same computer anyway!). Thus, on the real vehicle, Moe (the computer) digitizes video images and ships them over the ethernet to Curly. Curly digitizes laser range images and processes all of the video and range images and simulates the neural network. Larry supervises communications between computers, and may in the future perform the neural network simulation (if the communications delays become sufficiently small). Currently, this scheme allows Moe and Curly to begin digitizing their respective images simultaneously. The video image from Moe is ready before the range image is ready, and communications latency is kept to a minimum.

Speed of the system is important, and in order to keep the whole control loop going at slightly faster than a convenient if slightly arbitrary goal of 1 Hz, we must not only perform the parallel digitization, but we must reduced the images from their original higher resolution to lower resolutions which cut down on communications times, and - more importantly - cut down on processing time for the neural network simulations. Fortunately, in order to achieve the goal of proof of concept for the sensor fusion mission, we need not detect every obstacle which could be harmful to the HMMWV (which is impossible even at the highest resolution of the ERIM sensor when you consider the case of sharp metal or wooden poles pointed at the HMMWV which are below the sensing resolution). We count on the safety driver to keep the vehicle from harm, and try to detect only the larger obstacles autonomously, so we feel safe in this experiment in using ERIM images which are 16 x 64 pixels, and camera images which are 15 x 20 in each of the red, green, and blue bands (reduced from 64 x 256 and 480 x 640 respectively). This meant that most of our runs were stopped at some point when the vehicle encountered an obstacle it physically could not see. Clearly, faster computing and more efficient use of computing resources can be used to allow greater resolution.

3.2 Network architecture

Just as the images were constrained in resolution, the neural networks had to be tailored for execution in real-time. Generally, this was not a problem because the biggest implication for the networks was that the number of hidden units (see [Rumelhart86] for a full explanation), must be kept to as few as possible to accomplish the task. But if we think of artificial neural networks as function fitters, creating a mapping from the input to the output, the number of hidden units is roughly corresponding to the dimensionality of the function you try to fit to the data. If you try to fit a high dimensional function to a data set which could be represented better by a low dimensional function, you will fail to extrapolate from or (worse) interpolate between the provided data points. Thus, for a given training set of images and steering directions, you want to use a neural network with as few hidden units as necessary anyway.

3.2.1 Monolithic architectures used

For our purposes, we used two monolithic neural network architectures which performed almost identically. Most of the time we used a simple three layer feed-forward network with inputs from the ERIM and the CCD camera, five hidden units and eleven output units. From the above discussion on images, this means we have a network with 1860 inputs. The eleven outputs were chosen to be the same number of turning options that Kelly's Ranger system [Kelly93] has. The five hidden units used were based on results from Pomerleau [Pomerleau92] and from experimentation. The output activations were done in a gaussian form, such as Pomerleau used, so that the activation was highest on node corresponding to the desired steering direction and slope off on the nodes to either side (in a gaussian shape). On a few occasions an additional hidden unit of five nodes was added between the first hidden layer and the outputs in order to make the number and complexity of connections similar to those found in the MAMMOTH network used. See Figure 5, on page 10 for a view of this network.

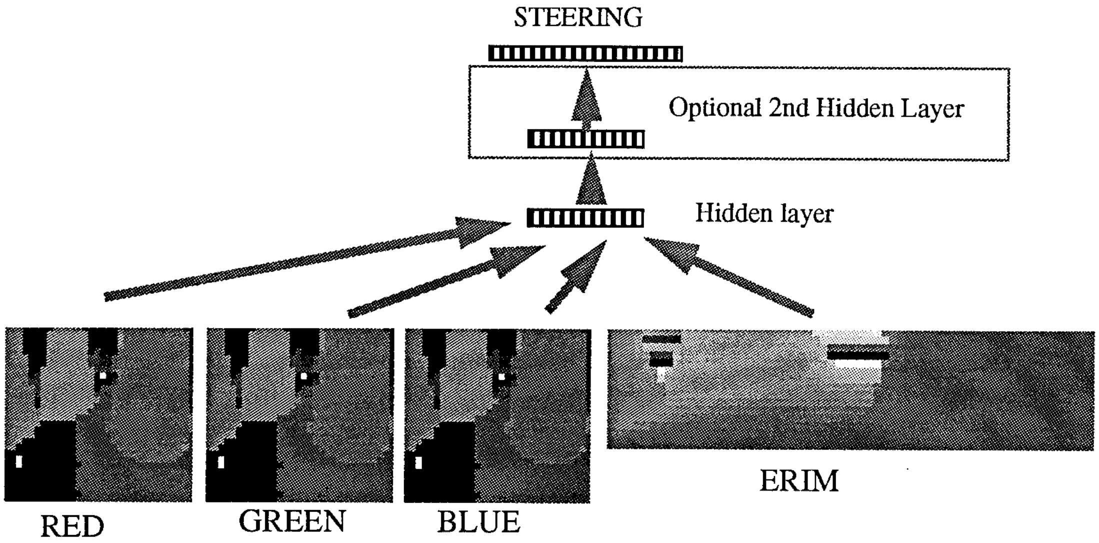

FIGURE 5. Monolithic navigation network (similar to ALVINN [Pomerleau92])

Training the monolithic network was straightforward for a given task. We simply gathered a training set of 300 exemplars (for our simple tasks this was sufficient). An exemplar consists of an input image and a corresponding desired output direction. Then the network was trained off-line. Usually this took two to five hundred epochs (or runs through the whole training set modifying the connections in the network each time) until it performed adequately for our tasks.

3.2.2 The specific MAMMOTH architecture

For all of our experiments we used the same MAMMOTH architecture, one which had the same number of inputs and outputs as the monolithic network, and which had roughly the same number of connections (since the number of connections corresponds directly to the processing time). It had the same number of inputs, but in this case, all of the CCD inputs (in red, green, and blue) are connected to one set of hidden units which are connected to one set of outputs. This feature network, or FeNN, was to be trained to find features in the color image space. Another similar FeNN connected the range image inputs to a second set of hiddens and to another output. This FeNN was for finding range image features. The task network with a set of hidden units and its own output layer sat above this and was connected to the hidden units of the two FeNNs. See Figure 6, on page 11, for a picture of the MAMMOTH architecture used.

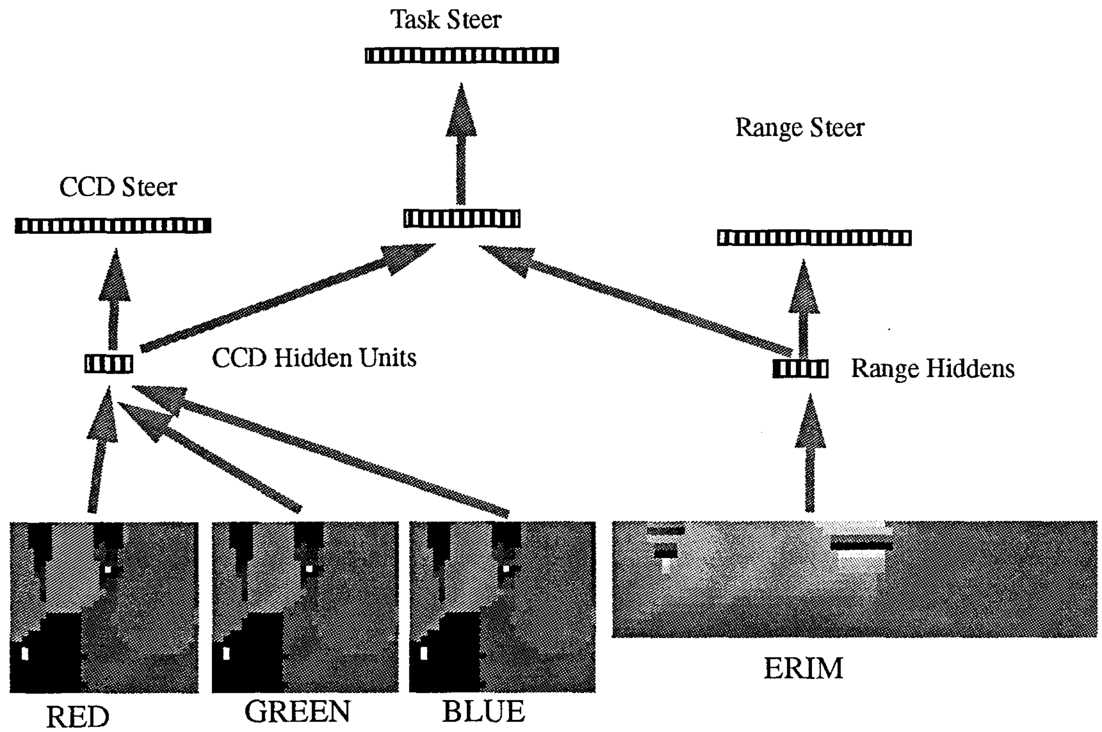

FIGURE 6. MAMMOTH network for navigation

The interesting aspect of the FeNNs is that they are trained before the task network is trained, and then their weights are frozen. The task network is then trained with the hidden units of the FeNNs as its inputs. If the FeNNs are well designed and the training sets for the FeNNs are constructed carefully, you can force these hidden units to represent a narrow portion of the input space. In [Davis93a] and [Davis93b], FeNNs were trained to find edges in an image in order to help a MAMMOTH network which located rivets on the skin of an aircraft. The edge FeNNs were trained entirely with artificially generated images, so they were constricted to find what we wanted to find. For all of the experiments covered in this work, the task network was trained with the exact same training sets as the monolithic networks.

4.0 Experiments and results

The experiments detailed here can be broken down into simulated and real world. The real world experiments were all done first in simulation, and the details of those results will only be mentioned as is relevant to the real world experiments, but additional simulation experiments were done in the simulator which had no direct analog in the real world, although they did have relevance, and those, too, will be detailed here.

4.1 Two interesting simulated experiments

The two experiments in the simulator which we shall review here involved taking the road following work of Pomerleau [Pomerleau92] a step further by adding obstacle avoidance through the addition of the laser range sensor. Pomerleau's experiments used just the video camera (in gray scale mode with shadow correction through the blue band) which is significantly faster than the 2 Hz ERIM, and this allowed him to achieve highway speeds. For our experiments, we had to slow down in our virtual world to accommodate the virtual ERIM, but we achieved a new level of capabilities. These new sensor fusion capabilities open new avenues in cross country navigation as well as road following.

4.1.1 Road with road-colored obstacles

The first experiment involves following a road while avoiding obstacles. In this case the obstacles were stationary because our simulator does not yet have the capability to handle moving obstacles, but since the system is reactionary and has no memory, it would treat stationary and moving obstacles similarly. In this first test, the obstacles were black, which, in the simulator, was exactly the same color as the road. The goal was to get the vehicle to follow the road, but avoid obstacles when it had to (see Figure 7, on page 13 for a map of the simulated terrain). The vehicle had to use both sensors (color video and laser range) to accomplish this goal.

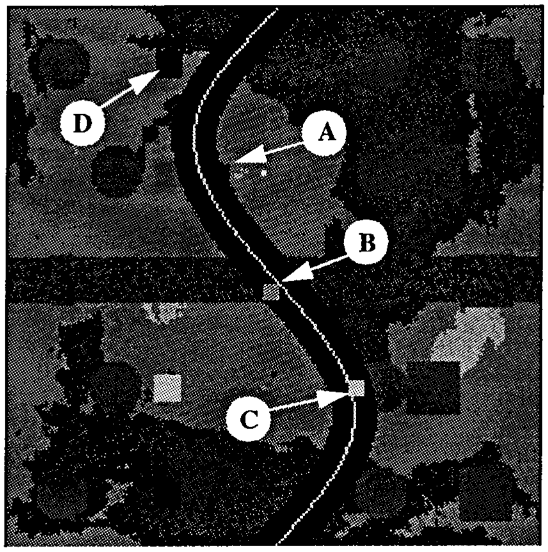

FIGURE 7. Map of the world with road and obstacles in the simulator. Block shaped obstacles are at points A, B, & C. A black pit is at point D. The road we want the vehicle to follow is the paved road with the yellow line. There is a "dirt" road (brown) running east-west, and various cylindrical and block-shaped obstacles off of the road.

Both network architectures were used for this test. For the monolithic network's training set and the Task Net of the MAMMOTH network, the vehicle was driven using the mouse to provide a steering direction over the road several times in each direction while collecting training exemplars. Additional exemplars were gathered by driving off of the road (with data collection off) and then driving back onto the road while collecting images and steering directions. These exemplars were necessary so that the simulated vehicle could recover from small navigational errors. All the while, the vehicle was steered so as not to hit any of the three obstacles on the road.

For the monolithic network we were able to train just with this data. However, for the MAMMOTH network, we had to train up the FeNNs. One FeNN was for road-navigational features using the simulated CCD camera and the other FeNN was for obstacle-avoidance-navigational features using the simulated laser range finder (look back to Figure 6, on page 11, for details of the MAMMOTH network).

The simulator gives us the power to easily make training sets for FeNNs. The nature of a FeNN is that we want it to learn internal representations (weight patterns) that represent a narrow and known set of features. In the simulator we can force a FeNN whose input is an image in a broad image space to learn only what we want by creating alternate worlds in the simulator whose only features in the appropriate image space are those we wish to learn. Thus to train a FeNN to recognize low level video features of roads appropriate for navigation and not to key off of extraneous image space features, we make a world in which the only feature is a road. Likewise, to train an obstacle avoidance FeNN, we make a world in which the only solid objects are large obstacles to be avoided. See for maps of the "road world" and the "cone world."

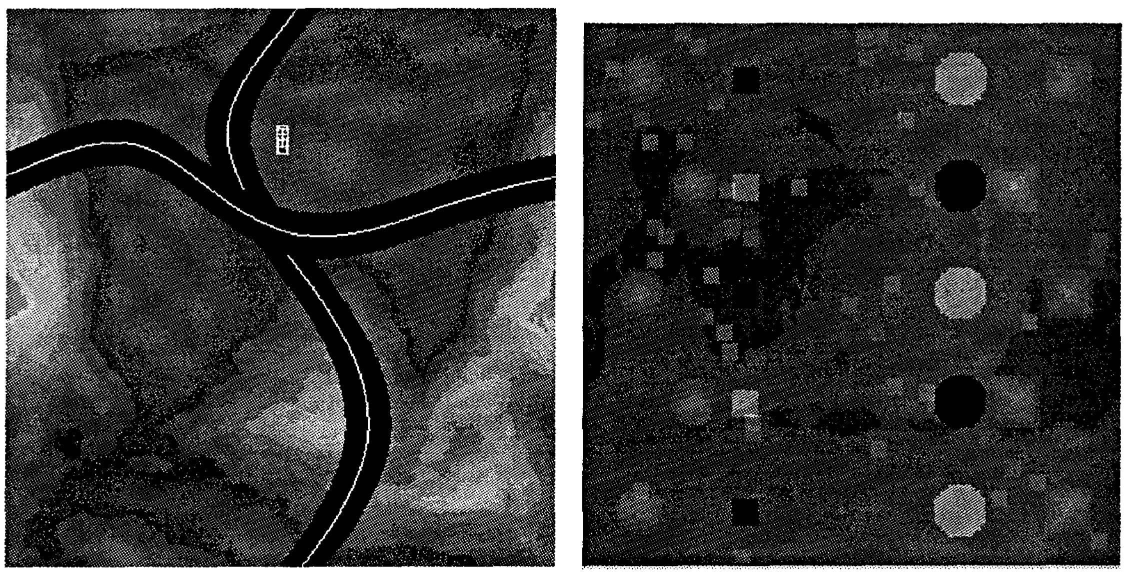

FIGURE 8. Road world (color map) and Obstacle World (depth map ~ light pixels are higher than dark pixels)

For the MAMMOTH network, 300 training exemplars were gathered in both the road-world and the obstacle-world. The respective FeNNs were trained with these and then the weights to the outputs of these FeNNs were frozen. Finally, the same training set used on the monolithic network was used to train the Task Net part of the MAMMOTH architecture.

Performance was excellent on this task for both networks. For both training and testing the speed of the simulated HMMWV was held perfectly constant, and at this constant speed the vehicle was able to stay on the road and avoid obstacles for over an hour and a half, which corresponds to about an hour at a slow speed [3mph] in the real world. Another option of the simulator allowed us to "warp" the vehicle to a random location and orientation in the simulated world, and three times out of four the vehicle resumed traveling on the road when it intersected the road again.

About every fourth time the vehicle would intersect the road at an almost right angle and that was sufficiently different from what was in the training set that it would respond improperly and drive straight across the road; with the addition of exemplars to the training set that could easily be avoided.

4.1.2 Road with brightly colored obstacles

A simple variant of the previous simulator task highlights one of the major pitfalls that often hinders researchers using neural networks, and the results suggest two partial solutions. In this task, the goal was again to stay on the road and to avoid the static obstacles also on the road, but this time the three obstacles were colored bright blue, green, and yellow from north to south along the road. Training sets were gathered in exactly the same manner as in the task of Section 4.1.1 on page 12, and training was done for the same amount of time (approximately 500 epochs - or times through the whole training set in batches of 300 - for each network to train).

The results were similar to those of the previous section. The particular training set for this task produced slightly sloppier driving habits in the simulated HMMWV, and this was evidence by the occasional (maybe 1 in 8 occurrence) of driving fairly widely around the blue box at point A and then heading straight into the pit at point D. This behavior is bad. However the explanation and fix are simple. The reason this happened is because the vehicle had not driven so far to the left on the road near point A before and thus had not seen the pit very well before. In this simulated world, the pit and the walls of the pit are black - the same color as the road. As the road turned back to the right, if the vehicle had just avoided the block at point A widely, the vehicle would see the black of the pit better than the black of the road and "lemming" itself into a bit of trouble. The fix is simply to add examples of avoiding pits to the training set.

A more interesting result came up when these networks (trained with the blue, green, and yellow blocks) were tested on a simulated world which was identical except for having three bright red blocks instead of three differently colored blocks (with the blocks at exactly the same locations). The first symptom was that the vehicle went off the road and into the pit much more frequently. With the monolithic network there was at least 2 in 3 chance of going off of the road, and with the modular network there was approximately a 1 in 3 chance. The implication is clear. The person training the network had correctly driven more closely around the blue box at point A than the yellow box at point C in the training. And both networks had learned to go left around blue boxes more tightly than around yellow boxes.

With the red blocks, the network drove much more loosely around the block at point A, and saw the pit that much more often.

The second symptom was that the vehicle frequently went off of the road at point B. There was an almost 100% occurrence with the monolithic network and at least a 50% chance with the modular network. Both of these symptoms have the same explanation: the yellow block in the first colored-block world was the only clock with a red component in RGB. The network learned to go wide around the yellow block since it had to go around it on the left to stay on the road and there was also a left turn immediately after the block on the road. This made the network go widely to the left of the block at point A and often to the left of the block at point B.

The simple fix is to use bigger training sets which have examples of blocks of every color on both sides of the road. But it is also interesting to note the slightly better performance of the modular MAMMOTH network. Truly, the training set for the task network was identical as that for the monolithic network, but it does seem to have helped that the lower level representations were separated into representations strictly for obstacles in laser range space and representations strictly for road following. By designing our FeNNs to represent what we want narrowly, we seem to have decreased the amount that we keyed off incidental features. Further experiments will be necessary to test this hypothesis, however.

4.2 Two equally interesting real-world experiments

Both of the real-world experiments described here address the most troublesome aspect of the current cross country navigation task: vegetation. If everything in the world was a rigid body, we would know to simply avoid everything, and we could do it [Kelly93]. However, sometimes we want to drive over the deformable obstacles (especially when we have no other decent route). The laser range finder alone cannot find vegetation, so we must use an additional sensor, in this case the color video camera.

4.2.1 The additive obstacle test

It should be noted that we don't always want to run over vegetation. Even if we do want to run over it, an important first step in checking to make sure that we are running over it intentionally is to convince ourselves that we really do see it properly. This first task is thus somewhat easier than the second task which we are starting to address. The first task has a simple goal: don't hit big solid obstacles and also don't hit vegetation. In contrast, the second task had this goal: don't hit big solid obstacles and also don't hit vegetation unless your only other choice is to hit big solid obstacles.

The first task is easier because anything that would be an obstacle in laser range sensor space alone will still be an obstacle in the task space. In the second task a large soft bush might appear to be an obstacle in the laser range sensor space, but we might actually have to consider it to be along the best route in the task space. We call the first task the additive obstacle task because the additional sensor only adds new things to avoid. We trained and tested at the slag heap described earlier. Much of the testing was in an area containing an access road which frequently had vegetation and or cliffs or mounds on one side, the other side, or both sides.

We used the same network architectures as for the simulated tasks in Section 4.1 on page 12. It should be intuitively clear, however, that generating appropriate training sets is significantly more difficult. For the Task Net part of the MAMMOTH network and for the monolithic network, we had to drive the HMMWV around the terrain we wished to wander about it. This would have been simple, except we still needed examples of recovering from mistakes, which meant we had to make mistakes (with data collection off), and mistakes are that much trickier in a five ton truck than in a simulated five ton truck. The procedure we used was the same (drive off of the road or too close to a mound or cliff and then recover to the right position), but the implementation was more difficult, or at least more hair-raising. Consequently, speed varied greatly, and the training set was slightly less rich in recovery examples.

But training the FeNNs was the real challenge. In the real world we cannot simple train in a world that contains the narrow set of features we wish to detect (most of the time). The laser range FeNN was trained to avoid obstacles, as it had been in the simulator. There were several large mounds of earth sufficiently far from other objects (including vegetation) which we used as our prototypical obstacles, and we trained simply by avoiding these mounds as we had done in the simulator.

We also wanted the CCD video camera FeNN to produce representations of vegetation, and this was more difficult. We trained a FeNN to drive around avoiding vegetation. To do this we drove around avoiding vegetation (sounds simple enough). In the real world, though, images containing vegetation often contain other things; additionally, vegetation in the real world is not nearly as homogeneous as any feature we might want to recognize in the simulated world. We had to try to develop a training set that contained a rich set of examples of the HMMWV steering away from all of the types of vegetation we might encounter.

Given these difficulties, the FeNNs we trained both worked fantastically well, and the monolithic network and the Task Net performed even better. In fact, our two main problem were that in the real world there are plenty of obstacles that simply don't show up at the resolutions we used in sensor space (such as sharp steel bars and jagged cinder blocks) and that there are plenty of culs-de-sac at the slag heap.

Both of the FeNNs would quite frequently allow the HMMWV to drive for runs of up to .6 miles before we ended up in a cul-de-sac or had something dangerous right in front of us that could not be seen at the low resolutions at which we used the sensors. Another failure mode was usually the result of uneven ground or territory not covered in the training sets for the FeNNs producing images that were largely dissimilar to all of those in the training sets for the respective FeNNS. The most frequent failure mode for the CCD FeNN was driving right up to a cliff that was the same color as the road. Likewise a common failure mode for the laser range FeNN was driving into low vegetation that got progressively more dense until we gradually appeared in an area where the vegetation was dense enough that no correct steering direction could be determined simply from range images.

Combining the two sensor modalities had the desired results. Typical runs of the HMMWV were around 1 mile or more before we encountered a cul-de-sac, a dangerous obstacle below sensor resolution, or a previously unseen situation (such as a tilted image from a tilted vehicle). The network sometimes was confused by low tan grass which was relatively sparse thatjust barely registered in the video images and could be placed into the same category as the jagged cinder block shards and metal pipes (i.e. obstacle below sensor resolution).

Both the monolithic network and the MAMMOTH network performed very well, and there is not enough data for a significant quantitative comparison. The one most interesting issue that we could address was that the FeNNs' training sets were gathered on an earlier day than the Task Net/monolithic training set (separated by a week). Most of this testing occurred in the Autumn when the vegetation and weather was changing most rapidly, and on the first day when we tested the fully trained networks, the MAMMOTH net seemed to outperform the monolithic network, but on subsequent outings, the environment was progressively more dissimilar to the environment when the FeNNs were trained (in particular, there was much less green vegetation). On these days the monolithic network, which was trained only with data from the later data collections, seemed to produce smoother and better steering commands.

4.2.2 The conflicting obstacle test

The final test to be discussed here is one which we are just beginning to addresson the real vehicle. The location of the test is a section of roughly 1600 square meters on our slag heap testing grounds which has numerous man-made mounds of earth spaced just a bit wider than is necessary for the HMMWV to drive through. The mounds of earth are approximately three feet high and five feet in diameter, and between them vegetation grows out of control. Our task is to drive the HMMWV successfully through this obstacle course. The tough part is that the laser range sensor generates images in which the vegetation looks as bad as the mounds of earth for the most part, so the video camera is needed to allow differentiation between safe and unsafe steering directions. However, in order to function outside of the most densely vegetated areas, we need the laser range camera for avoiding isolated mounds.

Once again our testing uses both the monolithic neural network and the MAMMOTH network. Generating test sets is time consuming and we have been fighting with the weather, but have managed a few outings. The CCD FeNN from the MAMMOTH network was trained with freshly generated images of the new test region with a focus on the heavily vegetated sections, the laser range FeNN was trained with the same exact training set used for the test in Figure 4.2.1, on page 16, and the Task Net was trained with a set that was a concatenation of the training set for the CCD FeNN and an equal number of exemplars from areas of less dense vegetation. This last training set was also used for training the monolithic neural network.

In the few outings we've had, the monolithic neural network was largely unsuccessful. The modular MAMMOTH network also experienced failures, but was able to go through the test region about 40% of the time. Three factors are primarily responsible for these results. First, both the CCD FeNN training set and the Task/Monolithic training set included data from the video camera, and the lighting conditions were slightly different in each training set and also neither training set had the same lighting as was present during the test runs. Second, navigation through this region required very slow and fairly variable speeds, which has two implications: a. the images had differing levels of speed induced distortion, and b. at slow speeds the vehicle's sensors clear an obstacle before the vehicle's wheels do, so we would be turning around an obstacle, and our reactive system would suddenly decide that there was no obstacle present and the wheels would hit the edge of the obstacle. The third factor was simply limited time on the vehicle due to inclement weather and hardware failures. Each of these problems is addressable, and Section 5.2 on page 21 suggests steps we shall take to overcome these difficulties.

5.0 Conclusions

5.1 Milestones

With this work we have achieved new capabilities in cross country navigation and sensor fusion, and have continued an exploration of an alternate neural network paradigm which produces more understandable and reusable neural networks.

For cross country navigation, we have added the ability to handle vegetation intelligently. By providing training exemplars of a person driving a vehicle through or around a vegetated area, we can train a neural network to come up with an appropriate mapping that will allow the vehicle to approximate the performance of the human trainer. We have succeeded in training a network to avoid vegetation and the work in simulation and the preliminary field work suggests that we can also train the neural network to choose to drive over vegetation when the other options are worse.

The implications of this work for sensor fusion in general is that we can feel confident to add new sensor modalities to a given task. A monolithic neural network is capable of learning to fuse the sensing modalities, but the MAMMOTH architecture gives us even more control over how to utilize the data provided in any given modality. Using appropriate FeNNs we can add some of our a priori knowledge of useful features and at the same time increase our ability to interpret how each sensing modality contributes to the task performance.

Although this paper makes no claims to examine the neural network issues in any depth, we feel that the MAMMOTH architecture will add several capabilities to neural networks for robotics. Primarily, we get the benefits mentioned above of integrating our knowledge into the networks through the FeNNS and achieving better understanding of the internal functioning of the networks. However, preliminary results hint that by tightly focussing the lower level internal representations of the network through the use of the FeNNs we may be able to avoid cross-modality interference that can result in poor performance such as keying off incidental features (such as the redness of a block in the simulated world). Also the FeNNs provide a nice paradigm for network re-use, as the FeNNs for one task can be inserted into the MAMMOTH network for another (such as the laser range FeNNs used in the real world tests).

5.2 Future Directions

This paper addresses quite a few issues, and it is our intention to explore the most pressing of these issues in greater depth and to publish those results as soon as they are available. These issues fall largely into two categories: those dealing with cross country navigation and those dealing with neural networks.

In getting a real robot to perform a task as complicated as cross country navigation, we are obliged to put whatever system we use through a battery of quantitative tests as well as the qualitative analysis presented here, and we must use both the qualitative and quantitative results to perform the system engineering which will allow the vehicle to make optimal use of it resources. Metrics which we are designing to measure the vehicle's performance with different steering controllers include closeness to obstacles, average run length, run smoothness, speed at which the vehicle can be driven, and, once we advance past wander mode and try to have the vehicle seek goals, how efficient the path to the goal is. System engineering issues include examining the value of using more possible steering directions in training (currently we discretize the steering direction to eleven possible steering angles), automatically extrapolating from training set images to get a fuller training set (by taking advantage of our knowledge of the sensor geometry to generate shifted images), taking best advantage of the multiple processors available to increase cycle rate for the steering controller, and finding the best sensing resolution for a task based on processing speed and safety.

The neural network issues which we intend to explore focus on performance analysis, and we are taking a two-pronged approach to this analysis. First, we are currently developing a system called ANGIE (Automatic Network Generator) which will allow use to train multiple networks at once by using all available workstations in our lab at night, and generate new variants on their architectures (specifically hidden unit architecture) either through exhaustive search, some kind of Genetic Algorithm, or any other desired search algorithm (using metrics of performance of the networks based on provided training and test sets). At the same time, we wish to topologically and set theoretically analyze the MAMMOTH and monolithic networks and the functions we want them to learn in order to best match the complexity of the function to the complexity of the network. People have analyzed the complexity of networks before, and we wish to expand on that by analyzing the complexity of the functions we want them to learn in real world robotics tasks.

This work will continue to focus on real world robotics and the complicated task of cross country navigation. It is our intention to continue the cycle of having the real world performance direct the theoretical work as well as the theory influencing the real world work.

6.0 Acknowledgments

This work has been supported in part by a National Science Foundation Graduate Research Fellowship. The equipment used is provided in part by DACA76-89-C0014, Topographic Engineering Center, Perception for Outdoor Navigation and DAAE07-90-C-R059, TACOM, CMU Autonomous Ground Vehicle Extension. Thanks go to all of the Robotics Institute at Carnegie Mellon University and the Field Robotics Center, and in particular George Mueller, Jeremy Armstrong, and Jim Frazier for keeping the vehicle working, Dean Pomerleau, Todd Jochem, and Al Kelly for keeping the software working, and Tony Stentz and Mel Siegel for keeping me working.

7.0 References

[Brumitt92] B. Brumitt, R. Coulter, A. Stentz. "Dynamic Trajectory Planning for a Cross-Country Navigator," *Proc. of the SPIE Conference on Mobile Robots*, 1992.

[Daily88] M. Daily, et al., "Autonomous Cross Country Navigation with the ALV," *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*: 718-726, 1988.

[Davis93a] I. L. Davis and M. W. Siegel, "Automated Nondestructive Inspector of Aging Aircraft," *International Symposium on Measurement Technology and Intelligent Instruments*, Huazhong University of Science and Technology, Wuhan, Hubei Province, People's Republic of China, October 1993.

[Davis93b] I. L. Davis and M. W. Siegel, "Visual Guidance Algorithms for the Automated Nondestructive Inspector of Aging Aircraft," *SPIE Conference on Nondestructive Inpection*, San Diego, July 1993.

[Fahlman91] S. E. Fahlman, "The Recurrent Cascade Correlation Architecture," Carnegie Mellon University Technical Report, CMU-CS-91-100, 1990.

[Gluck93] M. Gluck and C. Myers, "Hippocampal Mediation of Stimulus Representation: a Computational Theory," To appear in *Hippocampus* (in press 1993), 1993.

[Hampshir89] J. B. Hampshire and A. H. Waibel, "The Meta-Pi Network: Building Distributed Knowledge Representations for Robust Multi-Source Pattern Recognition," IEEE-PAMI, 1989.

[Jacobs91] R. A. Jacobs, M. L. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, 3:79-87, 1991.

[Jochem93] T. Jochem, D. Pomerleau, C. Thorpe, "MANIAC: A Next Generation Neurally Based Autonomous Road Follower," *Proc of International Conference on Intelligent Autonomous Systems* (IAS-3), Feb 15-18, 1993.

[Kelly92] A. Kelly, A. Stentz, M. Hebert, "Terrain Map Building for Fast Navigation on rugged Outdoor Terrain," *Proc. of the SPIE Conference on Mobile Robots*, 1992.

[Kelly93] A. Kelly, "A Partial Analysis of the High Speed Cross Country Navigation Problem," Carnegie-Mellon University Ph. D. Thesis Proposal, 1993.

[Langer93] D. Langer, J. K. Rosenblatt, M. Hebert, "A Reactive System For Off-Road Navigation," Carnegie Mellon University Technical Report, CMU-RI-TR-93-, 1993.

[Marra88] M. Marra, R. T. Dunlay, and D. Mathis, "Terrain Classification Using Texture for the ALV," Proceedings of the SPIE Conference on Mobile Robots, 1992.

[Meeden93] L. Meeden, G. McGraw, and D. Blank, "Emergent Control and Planning in an Autonomous Vehicle," *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 1993.

[Pomerleau92] D. Pomerleau, *Neural Network Perception for Mobile Robot Guidance*, Ph.D. Dissertation, Carnegie-Mellon University Technical Report CMU-CS-92-115, 1992.

[Pomerleau91] D. Pomerleau, "Neural network-based vision processing for autonomous robot guidance," *Proceedings of SPIE Conference on Aerospace Sensing*, Orlando, FL, 1991.

[Rumelhart86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Ed. MIT Press, 1986.

[Smieja92] F. J. Smieja and H. Mühlenbein, "Reflective Modular Neural Network Systems," Technical Report, German National Research Centre for Computer Science, March, 1992.

[Stentz93] A. Stentz, "Optimal and Efficient Path Planning for Unknown and Dynamic Environments," Carnegie Mellon University Technical Report, CMU-RI-TR-93-20, 1993.

[Wright89] W. A. Wright, "Contextual Road Finding With A Neural Network," Technical Report, Sowerby Research Centre, Advanced Information Processing Department, British Aerospace, 1989.